Q6

این کار امکان پذیر است، چون به راحتی می توان از درخت ترای درخت دودویی معادل ساخت. یعنی هر نود یک اشاره گر به چپ ترین فرزند خود و یک اشاره گر به اولین خواهر سمت راست خود داشته باشد.

در این روش، عملیات های حذف و درج و جست و جو چند برابر می شود. چون برای رسیدن به فرزندان هر نود، باید برادران نود بعدی را پیمایش کنیم.

در درخت ترای، هر نود، به اندازه ی حروف الفبا (انگلیسی مثلا) می تواند فرزند داشته باشد.

برای مثال، اگر بخواهیم کلمه ی trie را به درخت ترای اضافه کنیم،

اول، از فرزندان ریشه، فرزندی که مربوط به حرف t است را پیدا می کند، اگر وجود نداشت، این فرزند را به ریشه اضافه می کند و سپس همه ی حروف کلمه را به ترتیب، هر کدام به عنوان فرزند کلمه ی قبلی، اضافه می کند.

اما اگر t فرزند ریشه بود، اشاره گر میره روش و دنبال حرف بعدی، یعنی r میگرده. اگه r رو پیدا کرد، دنبال i میگرده

حالا بر فرض مثال، در درخت ما از قبل کلمه ی try ذخیره شده و بخاطر همین اشاره گر نسته تا اینجا بیاد. اما دیگه بعد از قr حرف i رو نمی تونه پیدا کنه. پس خودش به فرزندان r فرزندی با عنوان i را اضافه می کند و به i فرزند e را اضافه می کند و در آخر چون e انتهای کلمه است، روی آن برچسب انتها را می زند.

چون مثلا ممکن است ما کلمه ی tries را هم داشته باشیم. و اگر برچسب انتها روی e برای trie نداشته باشیم، این رشته همیشه به عنوان tries شناخته می شود.

حالا اگر بخواهیم داده ای را حذف کنیم، ابتدا باید آن را پیدا کنیم و بدانیم که در درخت موجود است، پس از ریشه ی درخت شروع می کنیم و حرف به حرف آن را جلو می رویم، باید دقت شود که در آخرین حرف حتما برچسب انتها وجود داشته باشد. اگر نداشت، یعنی این کلمه در درخت ثبت نشده است.

در مثال قبل، اگر بخواهیم trie را حذف کنیم، چون بعد از آن باز هم کلمه ی tries ادامه دارد، پس فقط کافیست برچسب انتها را از حرف e حذف کنیم.

و اگر بخواهیم tries را حذف کنیم، چون در ادامه اش، بعد از حرف s حرف دیگری نیامده و ادامه دار نیست، حرف s را حذف می کنیم و یک خانه مسیری که آمدیم را به عقب بر می گردیم، اگر به برچسب انتها رسیدیم، یا به جایی رسیدیم که نود مورد نظر به حروف دیگری اشاره می کند، عمل حذف را خاتمه می دهیم، وگرنه باز هم حرف به حرف حذف را ادامه می دهیم.